# 一：文件I/O

1. int open(const char \*name,int flags); <fcntl.h>
2. int open(const char \*name,int flags, mode\_t mode); <unistd.h>
3. int creat(const char \*name,mode\_t mode);
4. ssize read(int fd,void \*buf, size\_t len);
5. ssize write(int fd, const void \*buf ,size\_t len);
6. int fsync(int fd);
7. int fdatasync(int fd);
8. void sync(void);
9. int close(int fd);
10. off\_t lseek(int fd, off\_t pos, int origin);
11. ssize pread(int fd, void \*buf, size\_t count, off\_t pos); #define \_XOPEN\_SOURCE 500
12. ssize pwrite(int fd, const void \*buf, size\_t count, off\_t pos);
13. int ftruncate(int fd, off\_t len);
14. int truncate(const char \*path, off\_t len);
15. int select(int n,fd\_set \*readfds, fd\_set \*writefds, fd\_set \*exceptfds, struct timeval \*timeout);
16. FD\_CLR(int fd, fd\_set \*set);
17. FD\_ISSET(int fd, fd\_set \*set);
18. FD\_SET(int fd, fd\_set \*set);
19. FD\_ZERO(fd\_set \*set);
20. int pselect(int n, fd\_set \*readfds, fd\_set \*writefds, fd\_set \*exceptfds,const struct timespec \*timeout, const sigset\_t \*sigmask); <sys/select.h> #define \_XOPEN\_SOURCE 600
21. int poll(struct pollfd \*fds, usigned int nfds, int timeout); <sys/poll.h>
22. struct pollfd{int fd ;short events; short revents;}
23. int ppoll(struct pollfd \*fds, nfds\_t nfds, const struct timespec \*timeout, const sigset\_t \*sigmask); #define \_GNU\_SOURCE

# 二：缓冲I/O

1. FILE \* fopen(const char \*path, const char \*mode); <stdio.h>
2. FILE \*fdopen(int fd , const char \*mode);
3. int fclose(FILE\* stream);
4. int fcloseall(void); #define \_GNU\_SOURCE
5. int fgetc(FILE \*stream);
6. int ungetc(int c, FILE \*stream);
7. int fgets( char \*str, int size , FILE \*stream);
8. size\_t fread(void \*buf ,size\_t size, size\_t nr, FILE\* stream);
9. int fputc(int c, FILE\* stream);
10. int fputs (const char \*str, FILE\* stream);
11. size\_t fwrite(void \*buf, size\_t size, size\_t nr, FILE\* stream);
12. int fseek(FILE \*stream, long offset, int whence);
13. int fsetpos(FILE \*stream, fpos\_t \*pos);
14. void rewind(FILE \*stream);
15. long ftell(FILE \*stream);
16. int fgetpos(FILE \*stream, fpos\_t \*pos);
17. int fflush(FILE \*stream);
18. int ferror(FILE \*stream);
19. int feof(FILE \*stream);
20. void clearerr(FILE \*stream);
21. int fileno(FILE \*stream);
22. int setvbuf(FILE \*stream, char \*buf, int mode, size\_t size);
23. void flockfile(FILE \*stream);
24. void funlockfile(FILE \*stream);
25. int ftrylockfile(FILE \*stream);
26. 标准I/O的各种unlocked版本，需要定义\_GNU\_SOURCE

# 三：高级文件I/O

1. ssize\_t readv(int fd, const struct iovec\* iov, int count); <sys/uio.h>
2. ssize\_t writev(int fd, const struct iovec \*iov,int count);
3. struct iovec{void \*iov\_base; size\_t iov\_len;};
4. int epoll\_create(int size); <sys/epoll.h>
5. int epoll\_ctl(int epfd, int op, int fd, struct epoll\_event \*event);
6. struct epoll\_event{
7. \_\_u32 events; union{void \*ptr;int fd;\_\_u32 u32; \_\_u64 u64;}data;};
8. int epoll\_wait(int epfd, struct epoll\_event \*events, int maxevents, int timeout);
9. void \*mmap(void \*addr ,size\_t len, int prot ,int flags ,int fd, off\_t offset);
10. <sys/mman.h>
11. int munmap(void \*addr, size\_t len);
12. void \*mremap(void \*addr, size\_t old\_size, size\_t new\_size, unsigned long flags);
13. <unistd.h> <sys/mman.h> \_GNU\_SOURCE
14. int mprotect(const void \*addr ,size\_t len, int prot); <sys/mman.h>
15. int msync(void \*addr, size\_t len, int flags);
16. int madvice(void \*addr ,size\_t len, int advice);
17. long sysconf(int name); <unistd.h>
18. int getpagesize(void); 许多UNIX系统不支持
19. int posix\_fadvice(int fd, off\_t offset, off\_t len, int advice); <fcntl.h>
20. int readahead(int fd, off64\_t offset, size\_t count);

# 四：进程管理

1. pid\_t getpid(void); <sys/types.h> <unistd.h>
2. pid\_t getppid(void);
3. int execl(const char \*path, const char \*arg, …);
4. int execlp(const char \*file, const char \*arg, …);
5. int execle(const char \*path, const char \*arg, …, char \*const envp[]);
6. int execv(const char \*path, char \*const argv[]);
7. int execvp(const char \*file, char \*const argv[]);
8. int execve(const char \*path, char \*const argv[], char \*const envp[]);
9. pid\_t fork(void);
10. int exit(int status); <stdlib.h>
11. void \_exit(int status); <unistd.h>
12. void \_Exit(int status); <stdlib.h>
13. int atexit(void (\*function)(void)); <stdlib.h>
14. pid\_t wait(int \*status); <sys/types.h> <sys/wait.h>
15. int WIFEXITED(status);
16. int WIFSIGNALED(status);
17. int WIFSTOPPED(status);
18. int WIFCONTINUED(status);
19. int WEXITSTATUS(status);
20. int WTERMSIG(status);
21. int WSTOPSIG(status);
22. int WCOREDUMP(status);
23. pid\_t waitpid(pid\_t pid, int \*status, int options);
24. int waitid(idtype\_t idtype, id\_t id, siginfo\_t \*infop, int options);
25. pid\_t wait3(int \*status, int options ,struct rusage \*rusage); <sys/types.h><sys/time.h><sys/resource.h><sys/wait.h>
26. pid\_t wait4(pid\_t pid, int \*status, int options, struct rusage \*rusage);
27. int system(const char \*command); #define \_XOPEN\_SOURCE <stdlib.h>
28. int setuid(uid\_t uid); <sys/types.h> <unistd.h>
29. int setgid(gid\_t gid);
30. int seteuid(uid\_t euid);
31. int setegid(gid\_t egid);
32. int setreuid(uid\_t uid, uid\_t euid);
33. int setregid(gid\_t gid, gid\_t egid);
34. uid\_t getuid(void);
35. gid\_t getgid(void);
36. uid\_t geteuid(void);
37. gid\_t getegid(void);
38. pid\_t setsid(void);
39. int setpgid(pid\_t pid, pid\_t pgid); #define \_XOPEN\_SOURCE 500
40. int getpgid(pid\_t pid);
41. int chdir(const char \*path); <unistd.h>
42. int daemon (int nochdir, int noclose);
43. int dup(int oldfd); 将oldfd表示的文件描述符复制到当前进程可用的最小文件描述符。
44. int dup2(int oldfd,int newfd); 类似dup但是可以指定新的文件描述符为newfd

# 五：高级进程管理

1. int sched\_yield(void); <sched.h>
2. int nice(int nic); <unistd.h>
3. int getpriority( int which ,int who); <sys/time.h> <sys/resource.h>
4. int setpriority(int which ,int who ,int prio);
5. int ioprio\_get (int which ,int who);
6. int ioprio\_set (int which ,int who, int ioprio); 5和6都没有导出系统调用
7. int sched\_setaffinity (pid\_t pid ,size\_t setsize, const cpu\_set\_t \*set);
8. int sched\_setaffinity (pid\_t pid ,size\_t setsize, const cpu\_set\_t \*set);
9. void CPU\_SET(unsigned long cpu ,cpu\_set\_t \*set);
10. void CPU\_CLR(unsigned long cpu ,cpu\_set\_t \*set);
11. void CPU\_ISSET(unsigned long cpu ,cpu\_set\_t \*set);
12. void CPU\_ZERO( cpu\_set\_t \*set); 7-12:\_GNU\_SOURCE ,<sched.h>, typedef struct cpu\_set\_t , size\_t CPU\_SETSIZE
13. int sched\_getscheduler (pid\_t pid); <sched.h> policy: SCHED\_FIFO/RR/OTHER
14. int sched\_setscheduler (pid\_pid , int policy, const struct sched\_param \*sp);
15. struct sched\_param{ …int sched\_priority;…};
16. int sched\_getparam(pid\_t pid, struct sched\_param \*sp);
17. int sched\_setparam(pid\_t pid, const struct sched\_param \*sp);
18. int sched\_get\_priority\_min( int policy);
19. int sched\_get\_priority\_max( int policy);
20. int sched\_rr\_get\_interval( pid\_t pid, struct timespec \*tp);
21. int getrlimit( int resource ,struct rlimit\* rlim); <sys/time.h> <sys/resource.h>
22. int setrlimit( int resource ,const struct rlimit\* rlim);
23. struct rlimit{ rlim\_t rlim\_cur; rlim\_t rlim\_max; };

# 六：文件与目录管理