1.

```c
pid_t getpid(void);//pid
pid_t getppid(void);//parent_id
```

2.

```c
void exit(int status);///exit with code = status
```

3.

```c
pid_t fork(void);///parent process->son's pid;son process-> 0
```

4.

```c
pid_t wait_pid(pid_t pid, int *statusp, int options);
///pid > 0 wait pid
///pid = -1 wait all sons
///return the pid of process which is in waiting set {pid}(pid > 0),{all}(pid = -1)
pid_t wait(int *statusp);
///waitpid(-1, &status, 0);
```

5.

```c
unsigned int sleep(unsigned int secs);///wait for secs seconds,return the additional secon
int execve(const char *filename, const char *argv[], const char *envp[]);
///if failed return -1 else no return
```

6.

```c
char *getenv(const char *name);///search "name = value" -> return a pointer to value
int setenv(const char *name, char *newvalue, int overwrite);///if exist "name = oldvalue"
void unsetenv(const char *name);///delete "name = value"
```

7.

```c
pid_t getpgrp(void);///return process group
int setpgid(pid_t pid, pid_t pgid);///change ppid process to group pgid
///if pid = 0, use current pid
///if pgid = 0,the group pid = pid
```

8.

```
int kill(pid_t pid, int sig);/// send sig to pid
/// if pid > 0 -> to pid
/// id pid = 0 -> to each process of current process's group(including itself)
/// if pid < 0 -> to process group |pid|
unsigned int alarm(unsigned int secs);///after secs seconds send a sigalrm to process
```

9.

```
sighandler signal(int signum, sighandler_t handler);
///change signum's signal's action to handler
///error return SIG_ERR(without setting error)
```

10.

```
///default: when capture signal s,running handler of s,any other signal s is blocked,unti
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
///how = SIG_BLOCK:set signals in varible set to blocked(block |= set)
///how = SIG_UNBLOCK:block &= ~set
///how = SIG_SETMASK:block = set
///store previos to oldset(when oldset not null)
int sigemptyset(sigset_t *set);///initialize to empty set
int sigfillset(sigset_t *set);///add all
int sigaddset(sigset_t *set, int signum);///add
int sigdelset(sigset_t *set, int signum);///del
int sigismember(cosnt sigset_t *set, int signum);///if signum is a member of set
```

11.

```
int sigsuspend(const sigset_t *mask);///temporaly replace blocking set with mask until re
///after return restore it
///be used to wait a signal
```

12.

```
int setjmp(jmp_buf env);
int sigsetjmp(sigjmp_buf env, int savesigs);
void longjmp(jmp_buf ebv, int retval);
void siglongjmp(sigjmp_buf env, int retval);
```

13.

```c
int open(char *filename, 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);
ssize_t rio_readn(int fd, void *usrbuf, size_t n);
ssize_t rio_writen(int fd, void *usrbuf, size_t n);
static ssize_t rio_read(rio_t *rp, char *usrbuf, size_t n);
void rio_readinitb(rio_t *rp, int fd)
ssize_t rio_readnb(rio_t *rp, void *usrbuf, size_t n);
ssize_t rio_readlineb(rio_t *rp, void *usrbuf, size_t maxlen);
```