**Unix高级环境编程**

1. 每执行一个新程序，shell都会打开三个文件描述符：标准输入 标准输出 标准错误

#include <unistd.h>

STDIN\_FIFENO

STDOUT\_FIFENO

STDERR\_FIFENO

不带缓冲的I/O函数：open read write lseek close

char sBuffer[SIZE];

while ((n = read(STDIN\_FILENO, sBuffer, SIZE)) > 0)

{

    if (write(STDOUT\_FILENO, sBuffer, n) != n)

    {

        printf("write error \n");

    }

}

1. 按一个字符读取 getc(FILE \*) 输出字符 putc(int c,FILE \*) int ferror(FILE \*stream);检查错误

int c;

while ( (c = getc(stdin)) != EOF)

{

    if (putc(c, stdout) == EOF)

    {

        printf("write error");

    }

}

    if (ferror(stdin))

    {

        exit(0);

    }

fgets()一次读取一行

fork()创建子进程

1. 打印出错信息

#include <string.h> char \*strerror(int errnum)

#include <stdio.h> void perror(const char \*msg) //打印 msg: errno值的出错消息

1. 获取用户ID 组ID getuid() getgid()
2. 处理信号的三种方式：
   1. 忽略信号
   2. 按系统默认方式处理
   3. 提供一个函数，信号发生时调用该函数。

static void sig\_int(int)

{

    printf("get this signal \n");

}

int main(int arg, char \* args[])

{

    if (signal(SIGINT, sig\_int) == SIG\_ERR)

    {

        perror("signal error");

    }

    return 0;

}

1. 获取进程的时钟时间，用户时间，系统时间 time -p 进程名
2. 文件IO

不带缓冲指的是每个read和write都调用内核中的一个系统调用。

文件描述符的变化范围0~OPEN\_MAX。

#include <fcntl.h>

int open (const char \*path, int oflag, ...);

int openat (int fd, const char \*path, int oflag, ...);

若成功，则返回文件描述符；若失败，则返回-1。返回的文件描述符一定是最小未用描述符数值。

fd是通过打开相对路径名所在目录来获取。